Abstract

A method and corresponding equipment by which a synthesizer/ MIDI (musical instrument digital interface) device (10) is able to optimally perform a MIDI file (11) taking into account not the polyphony required by the MIDI file (11) as in SP-MIDI (scalable polyphony MIDI), but taking into account instead extended scalable polyphony (XSP) data 12b including the maximum number of instantaneous voices required by the MIDI file and the categories in which they occur for different channel masking, and also taking into account the architecture of the synthesizer/ MIDI device (10) in terms of a voice complexity coefficient table (12b) indicating the relative complexity (corresponding to a resource requirement) for voices in each category. The result is a total voice requirement table 12c-1 indicating typically less masking than would be required for the same synthesizer/ MIDI device to play the MIDI file according to SP-MIDI.

Appendix

Algorithm for the calculation of the total voice requirements based on MIV and corresponding classification data provided with a MIDI file. The code block for mip_length != miv_length allows backward compatibility with SP-MIDI; if MIV data is not provided with the MIDI file, then the algorithm mutes channels exactly as in SP-MIDI.

```
    Inputs
    polyphony:      The maximum number of Notes the player can play simultaneously
    max_capacity:   The maximum processing capacity of voices
    mip_length:     The number of entries in the MIP table
    miv_length:     The number of entries in the MIV table
    cla_width:      The number of architectures in the cla matrix
    mip[]:          A vector filled with MIP values
    miv[]:          A vector filled with MIV values
    cla[,]:         A matrix of the MIV value classifications for different
                    architectures
    cla_name[]:     A vector of the names of the architecture classifications in the
                    cla[,] matrix (cla_name[0] is used for unclassified voices).
    pri[]:          A vector of the MIDI Channel numbers in priority order
    n_ch:           Number of MIDI channels (16 for MIDI 1.0)

    Outputs
    mute[]:         A vector of 16 Boolean values specifying whether to mute the
                    corresponding MIDI Channel

    Functions
    Vcc(cla)        Returns the complexity coefficient corresponding to the voice
                    class

    Temporary variables
    i:              index variable
    j:              index variable
    ch:             Channel number
    vo:             voice counter
    to:             total voice requirement

for i := 1 to n_ch do
    mute[i] := TRUE
end for

if mip_length != miv_length then
    for i := 1 to mip_length do
        ch := pri[i]
        if mip[i] <= polyphony then
            mute[ch] := FALSE
        end if
    end for
else
    for i := 1 to miv_length do
        ch := pri[i]
        to := 0
        vo := 0
        for j := 1 to cla_width do
                vo := vo + cla[i,j]
            to := to + cla[i,j] * Vcc[cla_name[j]]
        end for
        to := to + (miv[i] - vo)* Vcc[cla_name[0]]
        if to <= max_capacity then
                mute[ch] := FALSE
        end if
    end for
end if
```